Disk allocation method:-There are three widely used allocation techniques Contiguous, Linked and indexed. The last two techniques are belong to non contiguous allocation.

1. Contiguous allocation:- In contiguous allocation files are assigned to contiguous areas of secondary storage. In this technique a user specifies in advance the size of the area needed to hold a file to be created. If the desired amount of contiguous space is not available the file cannot be created.

Advantage of contiguous allocation is that all successive records of a file are normally physically adjacent to each other. This increase the accessing speed of records.

2. Linked allocation :- In linked list allocation each file is linked list of disk blocks. This disk blocks may be scattered through the disc. A few bites of each disc block contains the address of the next block. The directory contains of pointer to the first and last blocks of the file.

Advantage of linked allocation is:

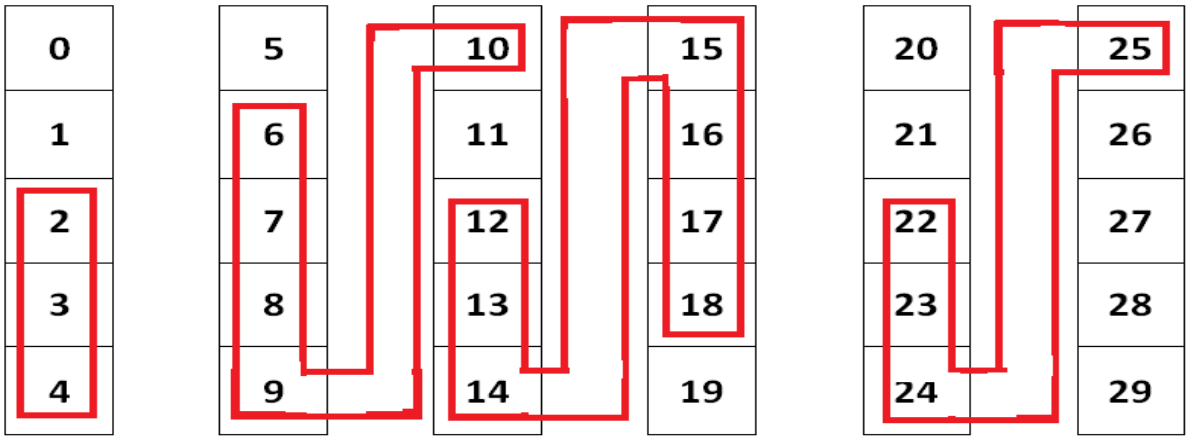
1. Simplicity
2. No disc compaction required:- Due to non contiguous nature of allocation, the linking does not produce any external disk fragmentation.

Disadvantage of linked allocation

1. Slow direct accessing of any disk block.
2. Extra space requirement for pointers.
3. Not reliable since disk blocks are linked by pointer a single damaged pointer can make thousands of disk blocks inaccessible.

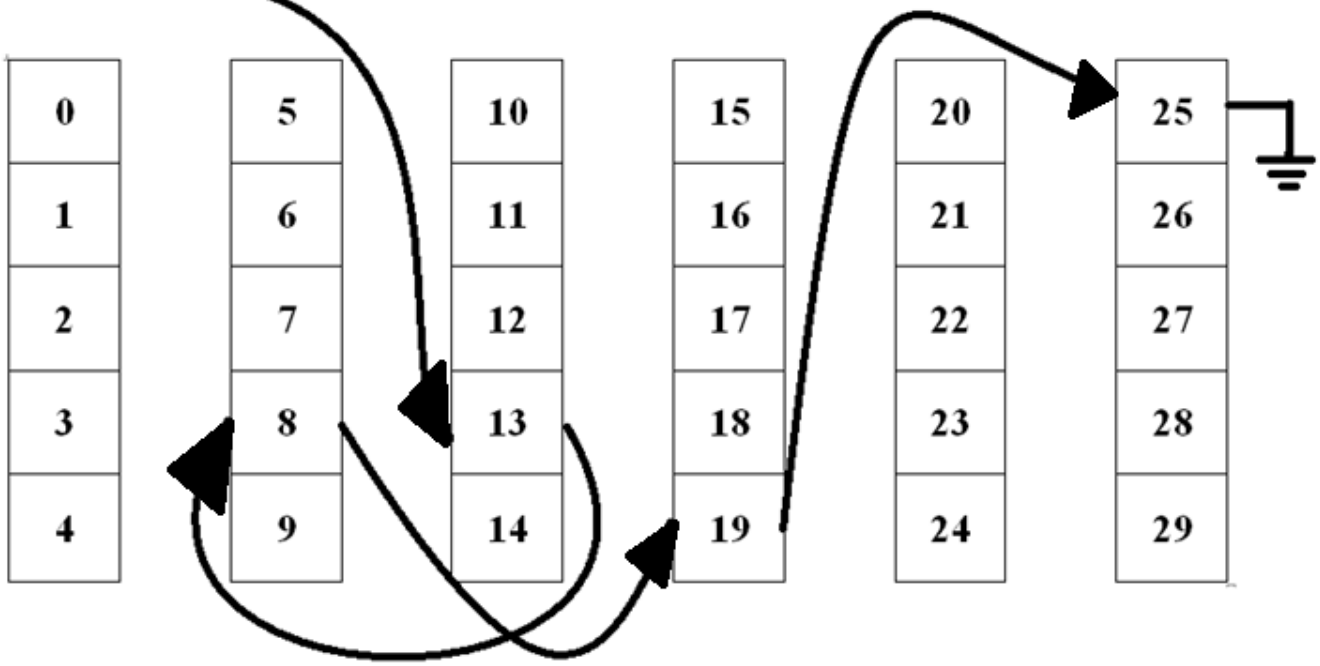
Indexed Allocation:- In indexed allocation scheme each file is provided with its own index block, which is an array of disk blocks pointers (addresses).

1. CONTIGUOUS ALLOCATION OF FILE



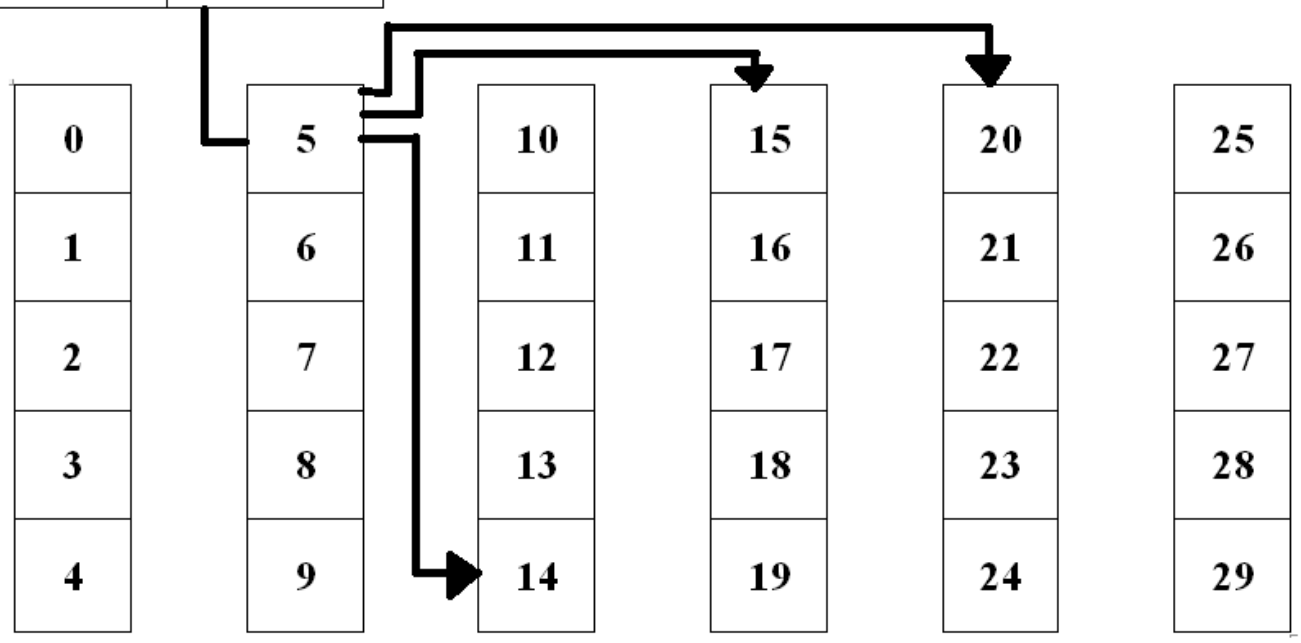
2. Linked Allocation

File	Start	End
F5	13	25



1. Indexed Allocation

File	Index Block
IGNOU	5



Disk scheduling

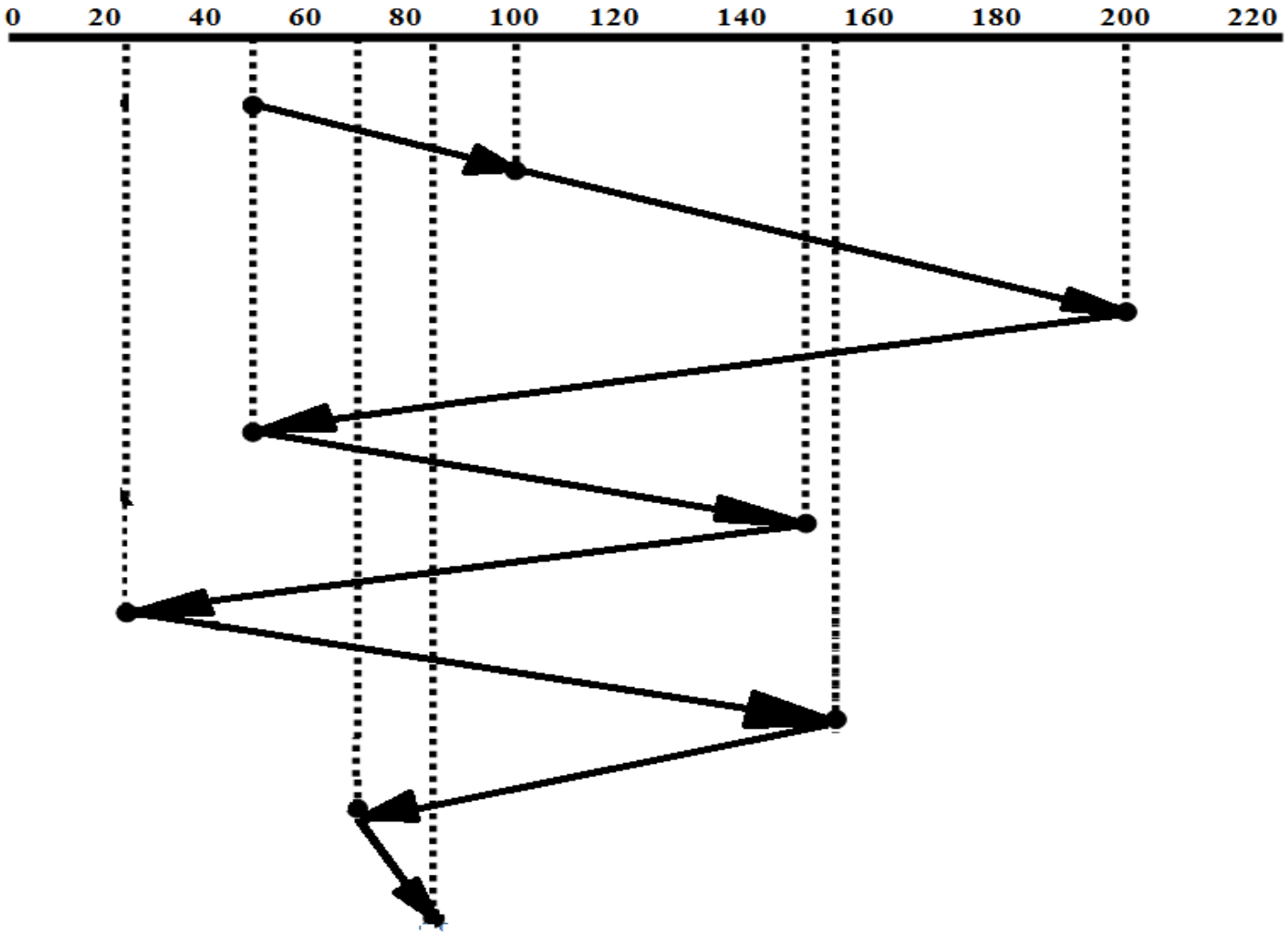
First come first served (FCFS) scheduling :-

This is the simplest form of scheduling in which the first request to arrive is the first one serviced FCFS scheduling has a fair policy in the sense that once a request has arrived, its place in the scheduling is fixed in the respective of arrival of a higher priority request. It may not provide the best service but it is easy to program. For example consider the following disk queue with request involve in track to record

100, 200, 50, 150, 25, 155, 70 and 85

The starting track to read is 100 and the last one is 85.

If the head position of a disk system is initially at 50, it will first move from 50 to 100 then to 200, 50, 150, 25, 155, 70 and 85 for a total head movement of 755 ((100 – 50) + (200 – 100) + (200 – 50) + (150 – 50) + (150 – 25) + (155 – 25) + (155 – 70) + (85 – 70)) tracks. The movement of a head for the track is illustrated in the following figure.



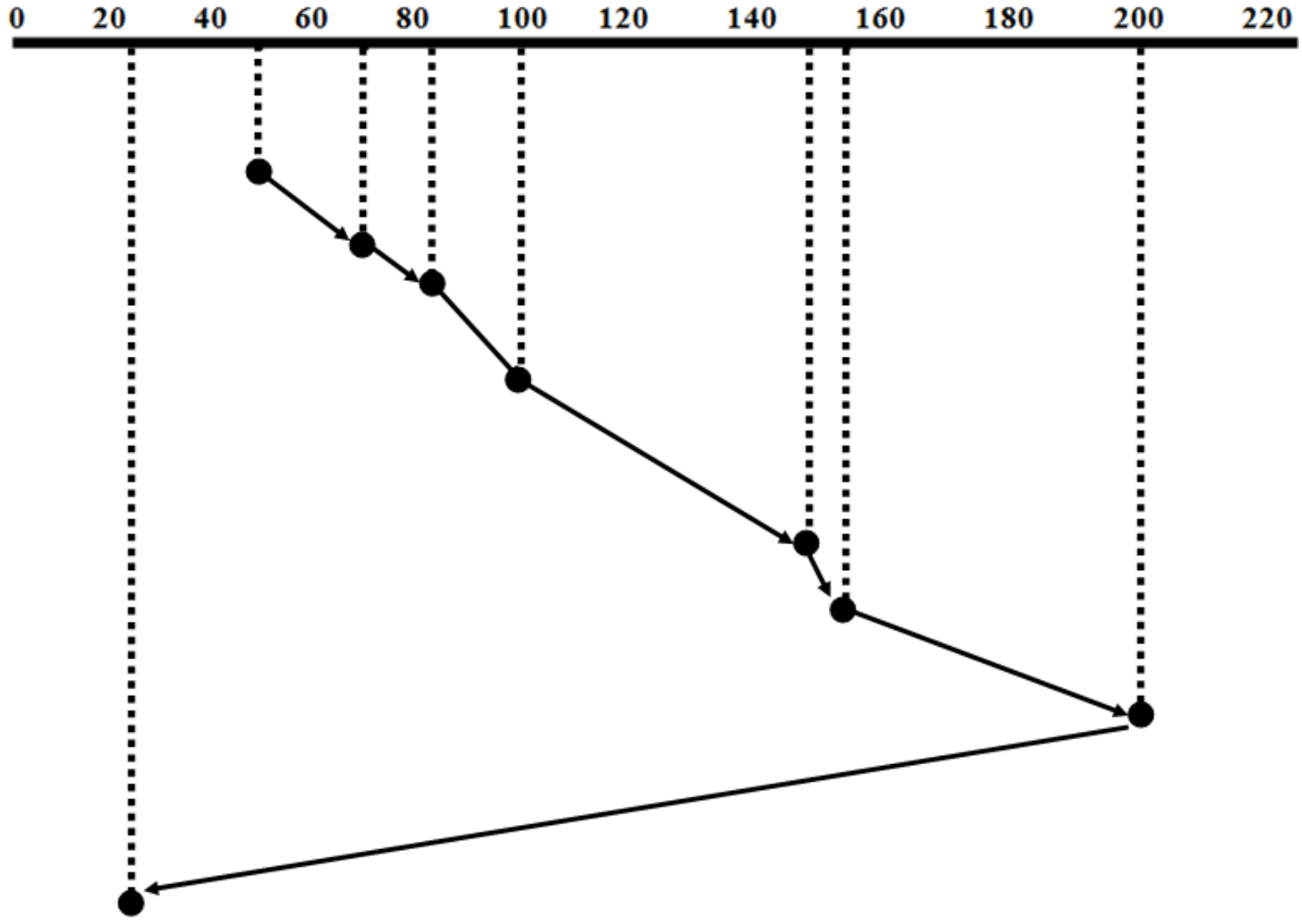
Drawback which this scheduling approach is a big swing as seen in the above figure from 150 - 25 and back to 155 FCFS is acceptable when the load on a disc is light

2. Shortest Seek Time First(SSTF) Scheduling:- In shortest seek time first scheduling priority is given to those processes which need the shortest seek, even if these requests are not the first one in the queue. It means that all requests nearer to the current head position are serviced together before moving head to distant tracks. Since Sick time is generally proportional to the track difference between the request, this approach is implemented by moving the head to the closest track in the request queue.

For example consider the previous example of request queue of 100, 200, 50, 150, 25, 155, 70 and 85 tracks, the nearest track for a service from an initial head position 50 is at track 70 once we are at 70 the next closest track is 85 (the difference is of 15 tracks only). From here it can go to 100 . Continuing this way the request for tracks 155 is serviced next and then switchover to 200 and finally at 25.

Movement of disk head:-

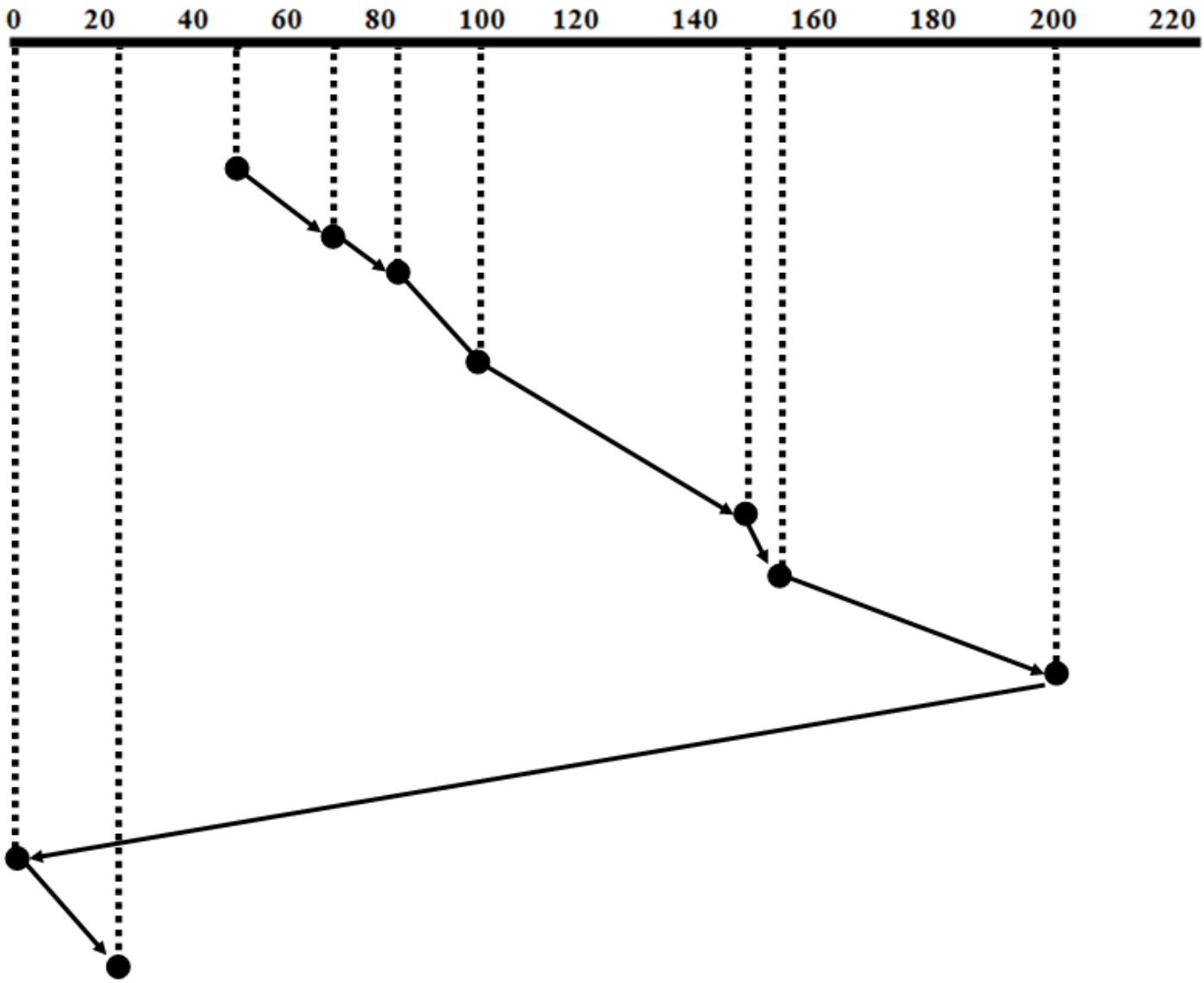
50 →70 → 85 → 100 →150 → 155 → 200 → 25



Thus using SSTF total head movement is 325((70-50) + (85-70) + (100-85) + (150-100) + (155-150) + (200-155) + (200-25)), which is less than a half of distance required for FCFS, with substantial improvement I disk throughput.

3. Scan scheduling:- In a scan scheduling the read or write ahead of the disc starts from one end moves towards the other end, services requests as it reaches each track until it reaches to another end of the disk. After reaching and other end of the disc, disk head reverses its path direction while continuing with services whichever comes on the way. This way disk head continuously oscillates from end to end. This is scheduling works with dynamic nature of request queue.

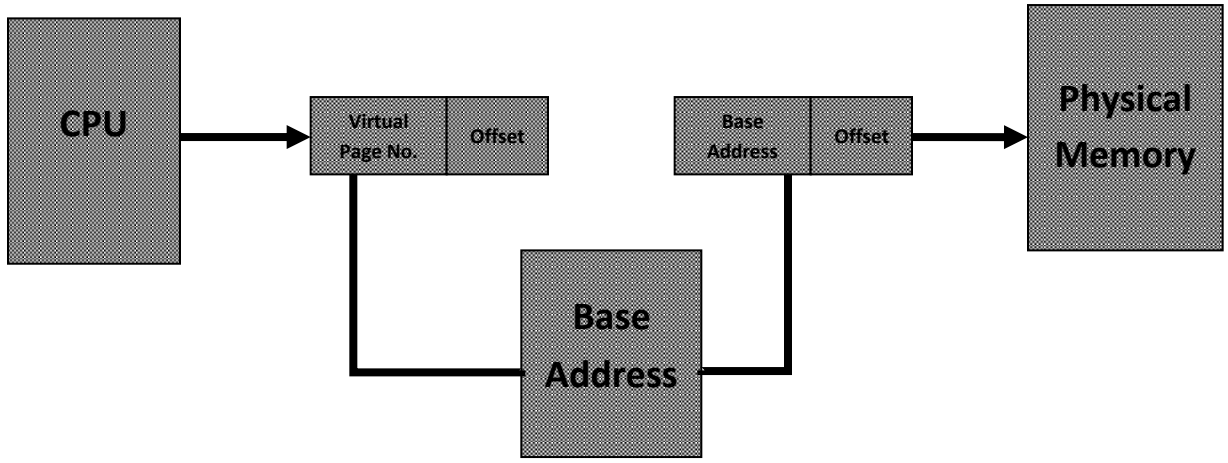
For example:- Suppose a disk head is moving from 0 and its last position was 45, then for the previous example 100, 200, 501 502 5155, 70 and 85 it would service 5070 85, 100 15015 5200 as it moves in that direction. If any request come on its way it will be serviced immediately while request arriving just behind the head will have to wait until disk head moves to the end of the disc, reverse is direction and returns before being serviced.



Paging:- Paging is a memory management technique that permits programs memory to be non-contiguous into physical memory thus allowing a program to be allocated physical memory wherever it is possible.

Address mapping in a paging system:- In this mechanism physical memory is conceptually divided into a number of fixed size blocks called frames (or page frames).The virtual address space or logical memory of a process is also broken into blocks of the same size called pages. When a program is to be run and its pages are loaded into any frame from the disc.

An important component of paging operation is a page map table(PMT), which contains starting address or base address of each page stored on physical memory



As shown in the figure, every address generated by CPU contains two components virtual (or logical) page number and a page offset into that page. The page number works as an index into the page map table. To define a physical memory address the base address in PMT is added with offset and sent to physical memory unit.

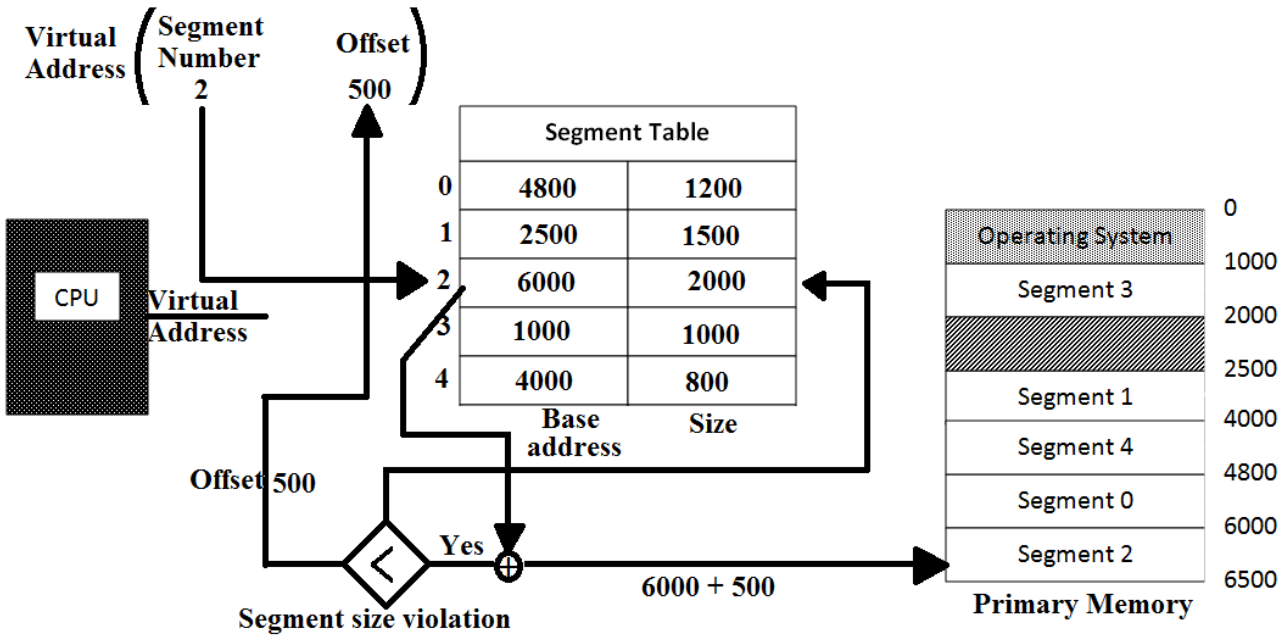
Segmentation:-

Segmentation is a memory management scheme which support programer's view of memory. Programmers never think of their program as a linear array of words rather they think of their programs as a collection of logically related entities, such as subroutines or procedures, functions, global or local data areas stack etc.

Segments are formed at program translation time by grouping together logically related entities. Formation of the segment vary one compiler to another.

Address mapping in segmented system:-

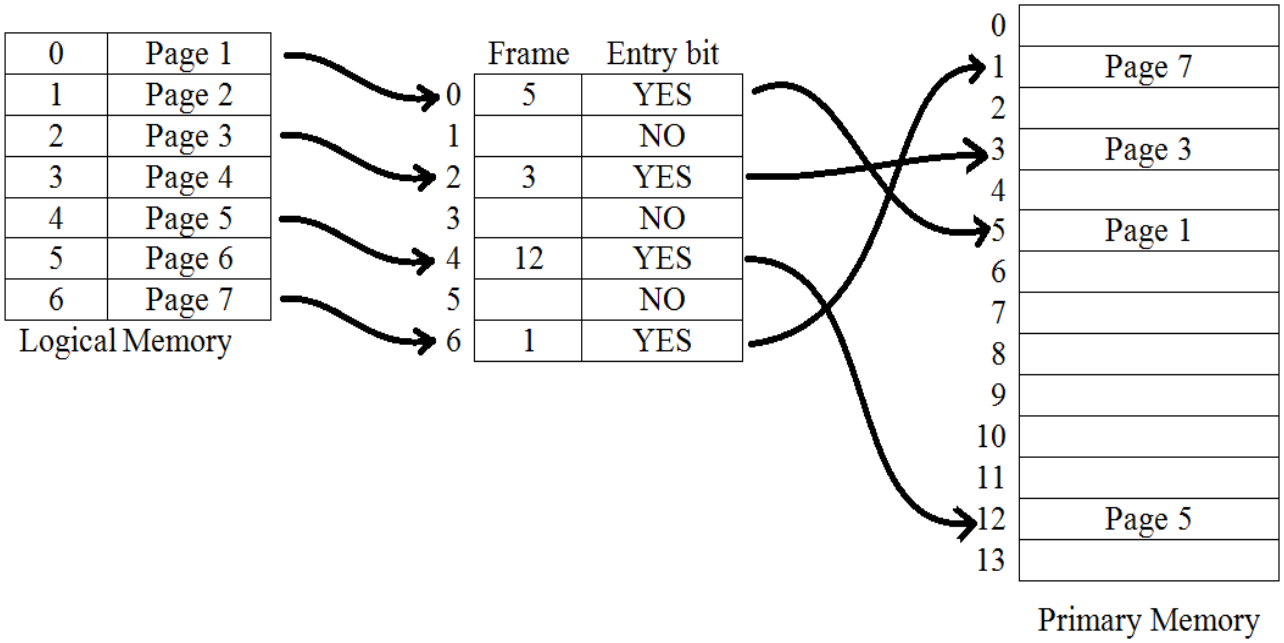
An important component of address mapping in a segmented system is a segment table. Its use is illustrated in the following figure.



A virtual (logical) address consists of two parts a segment number and an offset into the segment. The segment number provided in the virtual address is used as an index into the segment table. Each row the of the segment table contains a starting address (base address) of segment and a size of the segment. The address of virtual address must be within (less than or equal to) the size of the segment. If the offset of virtual address is not within the range it is trapped by the operating system otherwise the offset is added to the base address of the segment to provide physical address of the desired segment. Example considered the given figure there are 5 segments numbered from 0 to 4. The segment table has separate entry for each segment having the starting address (base address) of segment in physical memory and the size of that segment.

Demand Paging: In demand paging pages are loaded only on demand, not in advance. It is similar to paging system with swapping feature. Rather than swapping the entire program in memory only those pages are swapped which are required currently by the system.

To implement demand paging, it is necessary for the operating system to keep track of which pages are currently in use. The page map table contains an entry bit for each virtual page of the related process. For each page actual swept in memory, page map table points to actual location that contains the corresponding page frame and entry bit is set and marked as YES if it is in memory. Alternatively the entry bit is reset and marked as NO if a particular page is not in memory. If a program during execution never accesses those pages which are marked as NO, there will be no problem and execution proceeds normally. But if the program tries to access a page that was not swapped in memory page fault trap occurs.



Page Fault:- A page fault is a result of the operating systems failure to bring a valid part of the program into memory in an attempt to minimise swiping overhead and physical memory requirement. When the running program experiences a page fault, it must be suspended until the missing page is swapped in main memory. Here is a list of steps operating system follows in handling a page fault.

1. If a process refers to a page which is not in physical memory, then an internal table kept with a process control block is checked to verify whether a memory reference to a page was valid or invalid.
2. If memory reference to a page was valid, but the page is missing, the process of bringing a page into the physical memory starts.
3. Free memory location is identified to bring a missing.
4. By reading a disc, the desired page is brought back into the free memory location.
5. Once the page is in the physical memory. The internal table kept with the process and map table is updated to indicate that the page is now in memory.
6. Restarts the instruction that was interrupted due to the missing page.